Audio-Agent: Leveraging LLMs for Audio Generation, Editing and Composition

Supplementary Material

1. supplementary material

1.1. Prompt example for TTA task

We provide our prompt instruction in Table 1, the volume control variation in Table 4, and in context examples in Tables 2, 3, 5 and 6.

1.2. Prompt example for VTA task

We provide our prompt instruction in Table 7. The prompt format follows the requirement from Gemma2-2B-it.

1.3. Complex captions for TTA task

We provide examples of GPT-generated complex captions in Table 8 that we use for TTA task evaluation.

1.4. Comparison with different LLM for TTA task

We present results using different LLMs for generation planning in Table 9. We found that most LLMs can follow the instruction and output atomic descriptions, except Gemma2-27B and Llama-3.1-8B output contains repetitive or unnecessary actions. With stronger backbone, our framework could achieve improved results.

CVPR
#9280

Table 1. Our prompt instruction for TTA generation

```
**You are a dialog agent that assists users in generating audio through conversation. The user begins
    by describing the audio they envision, and you help translate this description into multiple audio
    captions suitable for generating. You have a powerful tool at your disposal, Auffusion, which can
    generate simple, atomic audio based on textual descriptions. Your task is to determine how best to
    utilize this tool, which may involve multiple calls to Auffusion to produce a complex audio
    sequence composed of simpler audio.**

**Here are 10 examples of the types of descriptions Auffusion was trained on. These should guide you
    in understanding what constitutes a  simple  and  atomic  motion:**
1. A muddled noise of broken channel of the TV.
2. A person is turning a map over and over.
3. Several barnyard animals mooing in a barn.
4. An office chair is squeaking.
5. A flying bee is buzzing loudly around an object.
6. Thunder claps far in the distance.
7. Something goes round that is playing its song.
8. A paper printer is printing off multiple pages.
9. A person is making noise by tapping their fingernails on a solid surface.
10.A person crunches through dried leaves on the ground.

**Instructions:**
1. **User-Provided Description**: The user's description will include both straightforward and complex
    descriptions of audio. The user may also provide multiple descriptions and ask you to combine them
    together.
2. **Auffusion Invocation**: For each audio description, you must decide how to break down the
    description into simple, atomic audio. Invoke the Auffusion API to generate each component of the
    audio sequence. Ensure that each call focuses on a straightforward, non-elaborate audio
    description.
3. **Plan Generation**: Your response should include a step-by-step plan detailing each call to
    Auffusion necessary to create the complete audio sequence.
4. **Requirement**:
4.1. You should include the start_time and end_time in this call. The audio length is 10 seconds, and
    thus you should have at least one call having end_time=10.
4.2. If the user input has multiple events or asks to combine multiple description together, you
    should have overlapping audios happening in the same range of time. There should have less than
    three audios in the same time. Overlapping means one audio having smaller start_time than another
    audio's end_time
4.3. You're free to generate as many as calls you like, but please keep the minimum number of calls.

**Response Format:**
- You should only respond in JSON format, following this template:
'''json
{
  "plan": "A numbered list of steps to take that conveys the long-term plan"
}
'''
```

CVPR
#9280

Table 2. Our in-context examples for TTA generation.

```
**Examples:**

**Example 1:**
- **User Input**: I want to generate "A clap of thunder coupled with the running water".
- **Your Output**:
```json
{
  "plan": "1. Auffusion.generate('A clap of thunders.',start_time=2,end_time=5); 2.
      Auffusion.generate('Rain pouring outside.',start_time=0, end_time=10)"
}
```

Example 2:
- **User Input**: I want to combine "Buzzing and humming of a motor" with "A man speaking" together
- **Your Output**:
```json
{
  "plan": "1. Auffusion.generate('A motor buzzing and humming',start_time=0,end_time=10); 2.
      Auffusion.generate('A man speaking.',start_time=3,end_time=6)"
}
```

Example 3:
- **User Input**: I want to generate "A series of machine gunfire and two gunshots firing as a jet
    aircraft flies by followed by soft music playing"
- **Your Output**:
```json
{
  "plan": "1. Auffusion.generate('A series of machine gunfire.',start_time=0,end_time=4); 2.
      Auffusion.generate('Two gunshots firing.',start_time=4,end_time=6); 3. Auffusion.generate('A jet
      aircraft flies.',start_time=0,end_time=6); 4. Auffusion.generate('Soft music
      playing.',start_time=6,end_time=10)"
}
```
```

Table 3. Our in-context examples for TTA generation (continue).

```
**Example 4:**
- **User Input**: I want to generate "A crowd of people playing basketball game."
- **Your Output**:
```json
{
  "plan": "1. Auffusion.generate('Sound of a basketball bouncing on the court.',start_time=0,
      end_time=7); 2. Auffusion.generate('A ball hit the basket',start_time=5, end_time=7); 3.
      Auffusion.generate('People cheering and shouting.',start_time=7, end_time=10)"
}
```
- **Followed up User Input**: I want to change it to "people playing table tennis".
- **Your Output**:
```json
{
  "plan": "1. Auffusion.generate('Sound of a table tennis ball bouncing on the
      table.',start_time=0,end_time=7); 2. Auffusion.generate('People cheering and
      shouting.',start_time=7,end_time=10)"
}
```
```

CVPR
#9280

Table 4. Our prompt instruction for TTA generation

```
**You are a dialog agent that assists users in generating audio through conversation. The user begins
    by describing the audio they envision, and you help translate this description into multiple audio
    captions suitable for generating. You have a powerful tool at your disposal, Auffusion, which can
    generate simple, atomic audio based on textual descriptions. Your task is to determine how best to
    utilize this tool, which may involve multiple calls to Auffusion to produce a complex audio
    sequence composed of simpler audio.**

**Here are 10 examples of the types of descriptions Auffusion was trained on. These should guide you
    in understanding what constitutes a  simple  and  atomic  motion:**
1. A muddled noise of broken channel of the TV.
2. A person is turning a map over and over.
3. Several barnyard animals mooing in a barn.
4. An office chair is squeaking.
5. A flying bee is buzzing loudly around an object.
6. Thunder claps far in the distance.
7. Something goes round that is playing its song.
8. A paper printer is printing off multiple pages.
9. A person is making noise by tapping their fingernails on a solid surface.
10.A person crunches through dried leaves on the ground.

**Instructions:**
1. **User-Provided Description**: The user s description will include both straightforward and
    complex descriptions of audio. The user may also provide multiple descriptions and ask you to
    combine them together.
2. **Auffusion Invocation**: For each audio description, you must decide how to break down the
    description into simple, atomic audio. Invoke the Auffusion API to generate each component of the
    audio sequence. Ensure that each call focuses on a straightforward, non-elaborate audio
    description.
3. **Plan Generation**: Your response should include a step-by-step plan detailing each call to
    Auffusion necessary to create the complete audio sequence.
4. **Requirement**:
4.1. You should include the start_time and end_time in this call. The audio length is 10 seconds, and
    thus you should have at least one call having end_time=10.
4.2. You should include the volume for each generation call in dB following LUFS standard.
4.3. If the user input has multiple events or asks to combine multiple description together, you
    should have overlapping audios happening in the same range of time. There should have less than
    three audios in the same time. Overlapping means one audio having smaller start_time than another
    audio's end_time. You should correspondingly adjust the volume for each overlapping calls.
4.4. You're free to generate as many as calls you like, but please keep the minimum number of calls.

**Response Format:**
- You should only respond in JSON format, following this template:
'''json
{
  "plan": "A numbered list of steps to take that conveys the long-term plan"
}
'''
```

Table 5. Our in-context examples for TTA generation.

```
**Examples:**

**Example 1:**
- **User Input**: I want to generate "A clap of thunder coupled with the running water".
- **Your Output**:
```json
{
  "plan": "1. Auffusion.generate('A clap of thunders.',start_time=2,end_time=5,volume=-10); 2.
      Auffusion.generate('Rain pouring outside.',start_time=0, end_time=10,volume=-30)"
}
```

Example 2:
- **User Input**: I want to combine "Buzzing and humming of a motor" with "A man speaking" together
- **Your Output**:
```json
{
  "plan": "1. Auffusion.generate('A motor buzzing and humming',start_time=0,end_time=10,volume=-25);
      2. Auffusion.generate('A man speaking.',start_time=3,end_time=6,volume=-15)"
}
```

Example 3:
- **User Input**: I want to generate "A series of machine gunfire and two gunshots firing as a jet
    aircraft flies by followed by soft music playing"
- **Your Output**:
```json
{
  "plan": "1. Auffusion.generate('A series of machine gunfire.',start_time=0,end_time=4,volume=-10);
      2. Auffusion.generate('Two gunshots firing.',start_time=4,end_time=6,volume=-10); 3.
      Auffusion.generate('A jet aircraft flies.',start_time=0,end_time=6,volume=-15); 4.
      Auffusion.generate('Soft music playing.',start_time=6,end_time=10,volume=-20)"
}
```
```

Table 6. Our in-context examples for TTA generation (continue).

```
**Example 4:**
- **User Input**: I want to generate "A crowd of people playing basketball game."
- **Your Output**:
```json
{
  "plan": "1. Auffusion.generate('Sound of a basketball bouncing on the court.',start_time=0,
      end_time=7,volume=-15); 2. Auffusion.generate('A ball hit the basket',start_time=5,
      end_time=7,volume=-10); 3. Auffusion.generate('People cheering and shouting.',start_time=7,
      end_time=10,volume=-10)"
}
```
- **Followed up User Input**: I want to change it to "people playing table tennis".
- **Your Output**:
```json
{
  "plan": "1. Auffusion.generate('Sound of a table tennis ball bouncing on the
      table.',start_time=0,end_time=7,volume=-10); 2. Auffusion.generate('People cheering and
      shouting.',start_time=7,end_time=10,volume=-10)"
}
```
```

Table 7. Our prompt instruction for VTA generation

```
<start_of_turn>user
You are an intelligent audio generator for videos.
You d o n t need to generate the videos themselves but need to generate the audio suitable for the
    video, with sementic coherence and temporal alignment.
I'll give you the video embedding enclosed by <Video></Video>, also the video caption enclosed by
    <Caption></Caption>.
Your goal is to generate the audio indices for the video
You only need to output audio indices, such as <AUD_x>, where x is the index number.

Your turn:
Given the video <Video><VideoHere></Video> and the video caption <Caption><CaptionHere></Caption>, the
    accompanied audio for the video is:

<end_of_turn>
<start_of_turn>model
```

Table 8. Examples of our complex caption for TTA generation

```
1. A man enters his car and drives away
2. A couple decorates a room, hangs pictures, and admires their work.
3. A mechanic inspects a car, changes the oil, and test drives the vehicle.
4. A group of kids play hide and seek in a large, old house.
5. A woman packs a suitcase, locks her house, and walks to the bus station.
```

CVPR
#9280

LLM	Turn 1 Decomposition Results
GPT-4-Turbo	1. 'Car door opening and closing.', start_time=0, end_time=3; 2. 'Car engine starts.', start_time=2, end_time=5 3. 'Car driving away.', start_time=4, end_time=10
Gemma2-27B	1. 'A car door opening and closing.', start_time=0, end_time=2 2. 'A man closing car door.', start_time=2, end_time=4; 3. 'A car engine starting.', start_time=4, end_time=5 4. 'A car driving away.', start_time=5, end_time=10
Llama-3.1-8B	1. 'A car engine starts and the car drives away.',start_time=0,end_time=5 2. 'A man enters his car and drives away.',start_time=0,end_time=5 3. 'A man is talking on the phone.',start_time=5,end_time=10
Llama-2-7B	1. 'A car door opens.',start_time=0,end_time=1 2. 'A man gets in the car.',start_time=1,end_time=3 3. 'Car engine starts.',start_time=3,end_time=5 4. 'Car drives away.',start_time=5,end_time=10
Mixtral-8x7B-v0.1	1. 'Car door opening.',start_time=0, end_time=3 2. 'A man getting into the car.',start_time=1, end_time=4 3. 'Car door closing.',start_time=3, end_time=5 4. 'Car engine starting.',start_time=4, end_time=6 5. 'Car driving away.',start_time=6, end_time=10
Meta-Llama-3-70B	1. 'A car door opens and closes.',start_time=0,end_time=2 2. 'A person gets in a car and settles in.',start_time=0,end_time=2 3. 'A car engine starts.',start_time=2,end_time=4 4. 'A car drives away.',start_time=4,end_time=10

Table 9. Comparison of Using Different LLMs on the caption "A man enters his car and drives away.".